(c) [1 point] $f(n) = \frac{n^2}{\sqrt[3]{n}}$ (i) $\frac{n^2}{\sqrt[3]{n}} \le \frac{n^3}{\sqrt[3]{n^3}} \le \frac{n^3}{n} \le n^2$ $\therefore O(n^2)$ ✗

(ii): $\frac{n^2}{\sqrt[3]{n}} \ge \frac{n^2}{\sqrt[2]{n}} \ge \frac{n^2}{\sqrt[2]{n}} \ge \frac{1}{\sqrt[2]{n}} \cdot n^2$ $\therefore \Omega(n^2)$ ✗ zero

(d) [1 point] $f(n) = (\log n)^{\log n}$ (i) $= (\log n)^{\log n} \le (\log n)^{\log n^2} \le \log n^{2 \log n}$

$\le 2\log n \log n \le 2\log 2n$ $\therefore \Theta(\log 2n)$

1

(ii): $(\log n)^{\log n} \ge$

(e) [1 point] $f(n) = n^{0.1}$ (i): $n^{0.1} \le n^{10} \le \log n^{10} \le 10 \log n$ ✗

$\therefore O(\log n)$

zero ?

(ii): n^0

Question 3 .... 7 points

Consider the following algorithm.

Algorithm 1 GE($A[0..n-1, 0..n]$) ▷ Input: An n-by-$(n+1)$ matrix $A[0..n-1, 0..n]$ of real numbers

```
for i ← 0, n − 2 do
    for j ← i + 1, n − 1 do
        for k ← i, n do
            A[j, k] ← A[j, k] − A[i, k] ∗ A[j, i]/A[i, i]
        end for
    end for
end for
```

(a) [1 point] Consider the multiplication as the basic operation. Find, in the worst case, the time efficiency class of this algorithm.

worst case: $n(n+1)$

time efficiency: $\Theta(n^2)$ ✗ zero

(b) [1 point] Consider the division as the basic operation. Find, in the worst case, the time efficiency class of this algorithm.

worst case: $\frac{n(n+1)}{2}$

time efficiency: $\Theta(n^2)$ ✗ zero

(c) [1 point] What evident inefficiency does this pseudocode contain?

A[i,K] * A[j,i]/A[i,i] in this code

we have to calculate it every loop ✓ 1

(d) [2 points] How can it be eliminated to speed the algorithm up?

{
tap = A[i,K] * A[j,i]/A[i,i]

for K ← i,n do

A[j,K] ← A[j,K] − tap }

✓ 2

(e) [1 point] How does this change affect the time efficiency of the algorithm in terms

the time efficiency will be

→ الصفحة التالية

of number of multiplications?

the time efficincy will be ?

(f) [1 point] How does this change affect the time efficiency of the algorithm in terms of number of divisions?

Question 4 .... *6 points*

Consider the following recursive algorithm for computing the sum of the first n cubes: $S(n) = 1^3 + 2^3 + ... + n^3$.

Algorithm 2 $S(n)$ ▷ Input: A positive integer n ▷ Output: the sum of the first n cubes

```
if n = 1 then
    return 1
else
    return S(n − 1) + n ∗ n ∗ n
end if
```

(a) [1 point] What is the algorithm's basic operation?

multiplications of n or Sumation of n

0.5

(b) [2 points] Set up and solve a recurrence relation for the number of times the algo-

تعويض

rithms basic operation is executed.

$T(n) = T(n-1)+2$

$T(n-1) = T(n-2)+2$

$T(n-2) = T(n-3)+2$

$T(n-3) = T(n-4)+2$

$T(0) = \cancel{1}^{0}$ $\therefore T(n-K) = T(0) = 1$

$n-K = 1 \rightarrow n = 1+K$

$\therefore O(n)$ ✓

2

$T(n) = T(n-1)+2 \rightarrow K=1$

$T(n) = T(n-2)+2+2 \rightarrow K=2$

$T(n) = T(n-3)+2+2+2 \rightarrow K=3$

$\vdots$

$T(n) = T(n-K)\cancel{+2K}$

(c) [2 points] Design a non-recursive version of this algorithm. Give a pseudocode of this algorithm.

no response

(d) [1 point] How does the recursive algorithm compare to the non-recursive one for computing $S(n)$?

no response